Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jure Potočnik

Aplikacija za preverjanje znanja z eQuizom na Android tablici

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: prof. dr. Aleksandar Jurišić

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Osrednji cilj diplomskega dela naj bo razvoj aplikacije za Android tablico, ki bo izvajalcem predmetov (učiteljem in asistentom) poenostavila administracijo pri izpitih. Oprla se bo na sistem eQuiz, ki ga razvijajo študentje FRI v okviru projektov e-Storitve oz. PKP. Posebej preučite varnostne probleme takšne aplikacije ter izberite ustrezne rešitve. Naredite tudi pregled uporabljenih orodij, programov in tehnologij, kot so Android, JavaScript, C#. Sledi naj podroben opis uporabe in delovanja same aplikacije ter dokumentacija njenega testiranja.

Pri izdelavi diplomske naloge bi se rad zahvalil celotni ekipi, ki je sodelovala na projektu in mi omogočila, da so zadeve stekle v skladu s pričakovanji. Zahvalil bi se študentu Andražu Dobnikarju, ki je pomagal pri API razvoju aplikacije. Posebej pa bi se rad zahvalil svojem mentorju, ki mi je pri izdelavi celotnega projekta vedno stal ob strani in pri zapletih rade volje priskočil na pomoč.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
HTML	Hyper Text Markup Language	Jezik za označevanje besedila
CSS	Cascading Style Sheets	Kaskande stilske podloge
AJAX	Asynchronous JavaScript and XML	Asinhroni JavaScript in XML
RAM	Random Access Memory	Pomnilnik z naključnim dostopom
JS	JavaScript	Javanski skriptni jezik
JSON	JavaScript Object Notation	Notacija objektov JavaScript
API	Application programming interface	Vmesnik za dostop do aplikacije
IDE	Integrated development environment	Integrirano razvojno okolje

Povzetek

Cilj diplomske naloge je izdelava Android aplikacije, ki bo nudila varen in enostaven način reševanja izpitov na tabličnih računalnikih. Program je razdeljen na dva večja dela. Prvi del programa deluje na Android tabličnih napravah in zagotavlja varno uporabo in prikaz vprašanj izpita. Uporabniku naprave onemogoča komunikacijo z zunanjim svetom in s tem uporabnik ne more pridobiti rešitev kvizov prek spleta. Drugi del aplikacije je narejen v ASP.Net kot spletna aplikacija in je namenjen pošiljanju in prejemanju podatkov o uporabnikih in kvizih med primarnim strežnikom in Android tablico.

Ključne besede: eQuiz, reševanje izpitov, Android aplikacija.

Abstract

Title: Application for examination with eQuiz on Android tablet

The purpuse of this thesis is to develop an Android aplication program which will provide a safe and easy to use system for solving exams on tablets. The program is devided into two parts. The first part of the program runs on the Android system which provides a secure way of showing the exams to the end users while preventing them of communicating with the outer world. The second part of the program runs on ASP.Net web application and is assigned to communicate with the primary server for sending and retriving data about the users and quizes.

Keywords: eQuiz, examination, Android application.

Poglavje 1

Uvod

Zadnje čase upada število asistentov na fakultetah in enako se dogaja tudi na naši fakulteti (FRI). Želja je, da imamo več vaj ter več sprotnega dela, s tem pa se veča želja po asistentih. Pri uvajanju več vaj in nalog, potrebujemo več asistentov, ki bodo te naloge popravljali. Na fakulteti smo razmišljali, kako bi lahko postopek popravljanja poenostavili ter pohitrili. Prišli smo na idejo, da bi ustvarili aplikacijo na tabličnem računalniku, s katero bodo lahko študentje reševali naloge, te pa bodo samodejno popravljene in ocenjene.

Na fakulteti se nas je zbrala ekipa študentov, ki smo bili pripravljeni izdelati spletno aplikacijo za izdelavo in popravilo študentskih nalog. Ekipa je začela z izdelavo projekta pod imenom eQuiz [7], ki ga vodi profesor dr. Aleksandar Jurišić. Moja naloga pri projektu je izdelati aplikacijo na tabličnem računalniku, ki bo študentom omogočila reševanje kvizov na preverjanjih znanja.

Cilj diplomske naloge je bila izdelava zaščitene aplikacije, ki se povezuje preko API-ja na oddaljen strežnik. Ta del projekta so razvili ostali študentje ekipe eQuiz, jaz pa sem poskrbel za pravilen prikaz izpitov in vprašanj. Aplikacija onemogoča prepisovanje nalog med študenti tako, da jim preprečuje preklope med programi. S tem študent izgubi možnost uporabe drugih aplikacij, ki so namenjene za komunikacijo z ostalimi uporabniki na spletu.

Sam izgled in uporaba aplikacije je zelo enostavna, saj je narejena tako,

da so gumbi na logičnih mestih in uporabnik takoj ve, kaj kateri gumb naredi. V primeru, da se mu zatakne pri prijavi, lahko tudi zahteva začasno geslo za prijavo v sistem.

Pred začetkom celotnega projekta smo pregledali, če že obstaja kakšna podobna rešitev naše naloge. Na internetu smo našli rešitev tEXAM [18], ki je v osnovi zelo podobna naši, a je celoten sistem malo drugačen. Pri naši aplikaciji lahko sistem po želji nadgrajujejo, je direktno integriran z internim informacijskim sistemom za izdelavo in popravilo izpitov, kot tudi s prijavo študentov v sistem.

Za večino problemov pri programiranju smo si pomagali s spletni stranjo StackOverflow, ki je med programerji zelo priljubljena. Na strani lahko postavimo vprašanja in odgovore. Vsako vprašanje in odgovor pa sta ocenjena s strani uporabnikov.

V 2. poglavju smo predstavili uporabljene programske jezike, v 3. poglavju smo predstavili, kako je arhitektura programa sestavljena in kako deluje povezava aplikacije z oddaljenim strežnikom. Opisan je tudi izgled celotne aplikacije in kako se lahko uporabnik po njej navigira. V 4. poglavju smo opisali varnost aplikacije, ki onemogoča prepisovanje odgovorov in vprašanj. V zadnjem poglavju, si lahko še preberete, kako smo se lotili testiranja aplikacije.

9 Answers

active oldest votes

35

You need two loops to implement the Bubble Sort .

Sample code :

```
public static void bubbleSort(int[] numArray) {

    int n = numArray.length;
    int temp = 0;

    for (int i = 0; i < n; i++) {
        for (int j = 1; j < (n - i); j++) {

            if (numArray[j - 1] > numArray[j]) {
                temp = numArray[j - 1];
                numArray[j - 1] = numArray[j];
                numArray[j] = temp;
            }

        }
    }
}
```

share edit flag

answered Apr 18 '13 at 17:01

NINCOMPOOP

36.1k ● 11 ● 89 ● 137

add a comment

Slika 1.1: Primer na spletni strani StackOverflow.com

Poglavje 2

Uporabljene tehnologije in programi

V poglavju smo omenili nekaj ključnih programskih jezikov, ki smo jih uporabili za razvoj aplikacije in opisali tudi, katere programe smo uporabili za razvoj.

2.1 Programski jeziki

Za razvoj aplikacije, ki deluje na android tabličnih napravah, je potrebno kombinirati kar veliko različnih programskih jezikov. Eni jeziki so namenjeni temu, da delujejo na samem android sistemu, drugi pa so namenjeni za uporabo pri programiranju spletnih strani.

HTML

HTML je jezik za oblikovanje spletnih strani. V sodobnem času jezik dopolnjujeta tudi CSS ter JavaScript. Jezik je sestavljen iz različnih gradnikov, ki jih imenujemo tudi HTML elementi. Njegov razvoj se je začel že v letu 1980, ko je fizik Tim Berners-Lee, takratni izvajalec v CERN-u, predlagal in prototipiral INQUIRE za uporabo in izmenjavo dokumentov. Kasneje je konec leta 1990 spisal prvi brskalnik ter strežnik, ki je uporabljal HTML.

```
▼<form action="/Login/LoginModel" method="post" autocomplete="off">
  ▼<div class="loginbox">
      <div class="title">Prijava na ekviz</div>
      <input autocomplete="off" id="Username" name="Username" placeholder="Username" type="text">
      <input autocomplete="off" id="Password" name="Password" placeholder="Password" type="password">
      <input type="button" value="Prijava" class="submit">
    </div>
  </form>
  <div class="copyright"></div>
```

Slika 2.1: Izgled HTML kode na spletni strani

Programski jezik HTML je na začetku vseboval samo nekaj osnovnih gradnikov, z njim pa smo lahko prikazali spletno stran, ki je vsebovala različne tipe pisav, slike ter tudi povezave do drugih spletnih strani. Vseboval je gradnike P, A, TITLE, HEAD ter LI in UL.

Gradnik P (Paragraph) označuje nov odstavek na spletni strani, Gradnik A nam omogoča ustvarjanje kazalca na drugo lokacijo na spletu. Gradnik TITLE se nahaja v HEAD gradniku in uporabniku prikaže naslov spletnega mesta. Gradnika, kot sta LI in UL, nam omogočata, da na spletni strani vstavimo oštevilčen in neoštevilčen seznam.

CSS

Ker je bil jezik HTML sam po sebi zelo monoton in z njim nismo imeli možnosti ustvariti sodobne spletne strani, je leta 1994 Håkon Wium Lie predlagal nov jezik, ki bi lahko na stran na enostaven način vpeljal različne barve, pozicije HTML elementov in podobno. Tako je leta 1996 izšel programski jezik CSS (Cascading Style Sheets). Pri jeziku je glavna prednost ta, da lahko njegovo kodo pišemo v novo, ločeno datoteko in se ne prepleta s kodo HTML. Na ta način dobimo boljšo preglednost med enim in drugim jezikom.

Pri jeziku CSS določamo različne sloge za elemente. Če se nek slog pojavi večkrat v dokumentu, potem imamo zapisana točna pravila, ki v kaskadnem načinu povejo, kateri slog ima višjo prioriteto. Pri uveljavitvi jezika so se pojavili tudi nekateri novi atributi, ki so postali del jezika HTML. To sta

atributa id (identifikator) ter class (razred). Z njima lahko vsak HTML element poimenujemo in se nanj sklicujemo iz ločene CSS datoteke. Pri tem moramo biti pozorni, da se na eni spletni strani lahko ime id atributa pojavi samo enkrat, za razliko od class atributa, ki se lahko pojavi večkrat.

Če se želimo iz CSS jezika sklicevati na id atribut HTML elementa, pred njegovim imenom dodamo znak #. V primeru, ko pa se želimo sklicevati na class atribut HTML elementa, pred njegovim imenom dodamo znak . (pika). V tem jeziku imamo možnost, da se sklicujemo tudi na gradnike HTML jezika. V tem primeru enostavno napišemo ime elementa brez, da bi pred njim postavili kakšen poseben simbol.

C#

Programski jezik C# se je prvič pojavil leta 2000. Razvilo ga je podjetje Microsoft, skupaj z ogrodjem .NET. Jezik je deklarativno, funkcijsko in objektno usmerjen ter vsebuje tudi možnost refleksije. V sedemnajstih letih razvoja programa smo prišli od prve verzije do verzije 6.0, ki je trenutno zadnja stabilna verzija.

Lojtra v imenu programskega jezika označuje dvakrat po dva plusa (++), simbola, ki sta del oznake jezika C++. V programskem okolju dva zaporedna plusa ponazarjata inkrementacijo spremenljivke za ena. Iz tega sledi, da je programski jezik C# nekako nadgradnja programskega jezika C++, ki je nadgradnja jezika C. Na sliki 2.2 lahko vidimo, kako deluje funkcionalnost ++ za povečanje indeksa v for zanki.

Jezik je spisan tako, da lahko dele njegove kode povežemo z različnimi operacijskimi sistemi. Zaradi tega lahko programe, ki jih prvotno napišemo v operacijskem sistemu Windows, zaganjamo tudi na operacijskih sistemih Linux z uporabo programa Mono.

```
@{
  var index = 1;
  Exercise question = null;
  foreach (var e in Model.exercises)
  {
    if (e.id == ViewBag.idExercise)
    {
      question = e;
      break;
    }
    index++;
  }
}
<div class="subHeader">
  @string.Format("{0} od {1}", index, Model.exercises.Count)
</div>
```

Slika 2.2: Primer povečanja indeksa za ena z okrajšavo ++

2.2 Tehnologije

Trenutno je Android najbolj popularen operacijski sistem za mobilne telefone. Razvilo ga je podjetje Google.

Android

Celoten sistem temelji na operacijskem sistemu Linux. Namenjen je predvsem napravam, občutljivim na dotik, kot so pametne ure in telefoni ter tablice. Dodatno se ga uporablja tudi za pametne televizije, avtomobile ter digitalne kamere in igralne konzole.

Razvoj sistema je v resnici začelo podjetje Android inc., a ga je leta 2005 kupilo podjetje Google. Prvo verzijo so izdali leta 2008. Najnovejša verzija Androida je izšla leta 2017 in je imenovana "Oreo"(piškot).

Ker so naprave z operacijskim sistemom Android običajno napajajo preko baterij, je sistem zasnovan tako, da skuša varčevati z energijo. Ko uporabnik zapusti program, sistem prekine delovanje z njim. V primeru, da želimo aplikacijo ponovno odpreti, jo imamo že v pomnilniku in je ni potrebno ponovno nalagati iz spomina telefona. Sistem sam zazna, ko mu začne primanjkovati

pomnilnika, in začne odstranjevati programe iz RAM spomina. Vrstni red odstranjevanja aplikacij je po principu prvi pride, prvi gre. To pomeni, da bo program, ki je prišel v pomnilnik prvi, tudi prvi odstranjen iz pomnilnika. Nekateri uporabniki Android telefonov nameščajo aplikacije, ki naj bi pravilneje ravnale s pomnilnikom telefona, a so naredili različne teste, kjer se je opazilo, da naredijo kvečjemu več škode kot koristi Vir: [16].

Android aplikacije se nahajajo na Googlovih strežnikih in jih lahko prenesemo preko aplikacije Google Play store. Trenutno je naloženih že več kot 2,7 milijona aplikacij. Mesečno aktivnih uporabnikov Android sistema je okoli dve milijardi.

API

Pri razvoju eQuiz aplikacije smo za komunikacijo med tablico in strežnikom uporabili API (Application programming interface), ki na podlagi podanih parametrov vrača podatke v naprej določeni obliki, ki jih uporabnik nato uporabi za prikaz na strani.

Pri celotni aplikaciji se v ozadju kličejo poizvedbe na glavni strežnik, ki ima dostop do podatkovne baze. Ko se uporabnik na tablici skuša prijaviti, aplikacija v ozadju naredi poizvedbo na strežnik z uporabniškim imenom in geslom, ki ga je vnesel uporabnik. Strežnik nato vrne podatke v JSON obliki, ki jih potem aplikacija uporabi za nadaljnje ukaze.

Na sliki 2.3 lahko vidimo odgovor strežnika po tem, ko smo nanj poslali poizvedbo o izpitih, ki jih lahko začnemo reševati. Vse podatke, ki jih odgovor vsebuje, lahko uporabimo za prikaz na zaslonu uporabnika ali pa jih uporabimo za pravilno delovanje aplikacije.

IIS

IIS (Internet Information Services) je spletni strežnik, ki ga je razvil Microsoft. Strežnik podpira veliko različnih protokolov, kot so HTTP, HTTPS, FTP, FTPS, SMTP in NNTP. Skoraj vse verzije strežnika so bile

```
{
  "id": 1,
  "examId": 1,
  "classId": 5,
  "isExam": true,
  "maxRetries": 1,
  "publishedAt": "2017-11-11T18:23:34.000Z",
  "startsAt": "2017-11-11 18:23:34",
  "finishesAt": null,
  "duration": 5400000,
  "data": null,
  "createdAt": "2017-11-11T18:23:34.000Z",
  "updatedAt": "2017-11-11T18:23:34.000Z",
  "passwordRequired": false,
  "class": "Team Red",
  "yearId": 6,
  "year": "2017",
  "groupId": 2,
  "group": "Mathematics",
  "title": "VIS 1. rok",
  "durationText": "an hour and 30 minutes",
  "durationMinText": "90 minutes",
  "timeLeft": 5400000,
  "timeLeftText": "an hour and 30 minutes",
  "timeLeftMinSecText": "90 min 0 sec",
  "timeLeftMinColonSecText": "90:00",
  "isOwner": true,
  "isSolving": false,
  "retriesCount": 0,
  "canStart": true,
  "statusText": "Exam ready to start.",
  "type": "Exam",
  "typeLower": "exam",
  "startedAt": null
},
```


Slika 2.3: Primer JSON odgovora o izpitu pri predmetu VIS iz skupine matematika. Vidi se podatke o imenu izpita, leto izdelave, čas trajanja reševanja in podobno.

nameščene skupaj s sistemom Windows. Strežnik podpira tudi WebDAV, ki nam omogoča oddaljeno posodobitev spletnih strani. IIS nam je na voljo že od leta 1995 in ima trenutno okoli 43% tržnega deleža [19].

IIS omogoča veliko različnih sistemov za avtentikacijo uporabnikov:

- anonimna avtentikacija,
- osnovno preverjanje identitete,
- preverjanje z uporabniškim imenom in geslom,
- preverjanje z obstoječimi uporabniki v sistemu Windows,
- UNC preverjanje [20],
- preverjanje s digitalnim potrdilom.

2.3 Uporaba programov

Pri programiranju bolj zapletenih aplikacij si radi izberemo urejevalnik, na katerega smo najbolj navajeni ali pa je enostaven za uporabo. Pri izdelavi smo si zbrali kar nekaj različnih programov, saj vsak po svoje zelo pomaga pri programiranju. Tukaj govorimo o preglednosti kode do urejevalnikovih priporočil, kako kodo izboljšati oziroma jo optimizirati.

Microsoft Visual Studio

Eden izmed IDE (Integrated Development Enviroment) razvojnih orodij podjetja Microsoft je Microsoft Visual Studio. Program nam omogoča programiranje spletnih aplikacij, namiznih programov, kot tudi mobilnih aplikacij. Visual Studio nam poleg urejevalnika kode nudi tudi IntelliSense, ki nam pomaga, da hitreje pišemo programsko kodo. Ko začnemo pisati ime spremenljivke, nam priporoča že uporabljeno ime, ki ga lahko izberemo s pritiskom na tipko Enter. Program za pisanje programske kode ima integriran tudi razhroščevalnik za odkrivanje napak pri programiranju. S programom se

lahko povežemo tudi na oddaljene programe in jih na daljavo pregledujemo in popravljamo. Program omogoča tudi nameščanje dodatkov, s čimer lahko program povežemo z Git ali SVN repozitorijem.

IDE podpira veliko različnih programskih jezikov, kot so C, C++, C#, F#, in tudi spletne programske jezike, JS, CSS, HTML.

Android Studio

Android Studio je uradni IDE urejevalnik kode za Android. Narejen je na osnovi JetBrains IntelliJ IDE programa. Program je zamenjal stari urejevalnik Android kode, Eclipse. Android studio je podprt na večini operacijskih sistemih, kot so Linux, Windows ter IOS. Poleg samega urejevalnika kode ima zraven integriran tudi program za avtomatsko zaključevanje kode. Kot dodatek k urejevalniku lahko na računalnik namestimo tudi emulator, ki se lahko poveže s programom. Z emulatorjem lahko simuliramo mobilne telefone na samem računalniku. S tem lahko pregledamo, kako bo izgledala aplikacija na različnih telefonih, brez potrebe po ročni namestitvi programov na vsako napravo posebej.

Poglavje 3

Razvoj aplikacije eQuiz

Kot smo že omenili v uvodu, je bil cilj izdelati aplikacijo, ki bo študentom omogočala reševanje izpitov na tabličnih računalnikih namesto na papirju. Pred prvo uporabo je potrebno tablico aktivirati v sistem, da pridobi dovoljenje za komunikacijo s spletnim delom programa. Po uspešni aktivaciji se lahko študent s svojimi vpisnimi podatki na njej prijavi ter opravlja izpit.
Celoten projekt je obsegal:

- spletno rešitev,
- API dostop do podatkov,
- postavitev strežnika za komunikacijo,
- aplikacijo za tablične računalnike,
- zaščito aplikacije pred zlonamerno uporabo.

Pri razvoju aplikacije smo se odločili, da jo razbijemo na dva dela. Prvi del sestavlja Android program, ki bo skrbel za zaščito pred morebitnim študentovim goljufanjem, drugi del pa bo namenjen študentu za interakcijo s sistemom.
Program, ki ga bo uporabljal študentom se deli na:

- prijava v eQuiz,

- prijava na izpit,
- reševanje podvprašanj izpita,
- pregled rešenih vprašanj,
- oddaja odgovorov,
- zaključek izpita.

Pred začetkom dela smo od mentorja pridobili dve tablici, na katerih smo lahko razvoj aplikacije testirali.

3.1 Android aplikacija

Aplikacija, ki deluje na Android sistemih, je morala skrbeti, da uporabnik ne more aplikacije zapreti, odpreti nove ali pa imeti možnost komunicirati z ostalimi uporabniki na spletu. Ko učitelj ali asistent aplikacijo prvič odpre, jo mora potrdili v sistemu, da je pooblaščena za reševanje izpitov. Ključ, ki ga aplikacija dobi od strežnika, se shrani na samo tablico, nato pa je uporabljen za vse nadaljnje komunikacije s strežnikom. Oseba, ki je tablico pooblastila za uporabo, jo lahko kadarkoli prek spletnega vmesnika tudi blokira. V tem primeru tablica v trenutku izgubi dostop do strežnika in reševanje izpita se takoj prekine.

Pri izdelavi aplikacije smo se odločili, da uporabimo tehnologijo WebView, ki nam je omogočila, da je program spremenila v nekakšen spletni odjemalec. Pri tem se je poenostavil razvoj aplikacije, vse nadaljnje posodobitve aplikacije pa lahko uredimo prek spleta, saj ni potrebno vsako tablico ponovno posodobiti.

Opis naprave

Od mentorja smo pred začetkom razvoja aplikacije dobili dve tablični tablici. Obe sta znamki Charm Intex in vsebujeta 512 MB RAM pomnilnika, 4GB notranjega pomnilnika ter 7” zaslon.

Slika 3.1: Prikaz strani za začetek reševanja izpita

WebView

V Android sistemu poznamo poseben pogled programa, ki se imenuje WebView. Z njim lahko izdelamo svoj brskalnik ali pa enostavno prikažemo neko drugo spletno stran. Za generiranje spletne strani uporablja WebKit. Slednji nam omogoča tudi prehod med prejšnjo in naslednjo stranjo brskanja z uporabo tipk na tablici. Pri izdelavi je bilo potrebno razširiti že implementiran WebViewClient, saj ta zaenkrat še ne podpira navigiranje na druge strani prek povezav na spletni strani.

3.2 Spletni del aplikacije

Drugi del diplomske naloge predstavlja izdelava spletne strani, na katero se Android program povezuje. Ko uporabnik prvič pride na spletno stran, mu ta priredi unikatno oznako ter odpre prijavno okno. Uporabnik se mora prijaviti s svojo vpisno številko in geslom. V primeru, da se uporabniško ime

in geslo ne ujemata, sistem uporabnika na to opozoril. Po uspešni prijavi ima študent na voljo izbiro izpita. Pri vsakem izpitu je naveden ime ter čas trajanja reševanja izpita. Ko uporabnik izpit izbere, se naredi ponovna poizvedba na strežnik in mu prikažemo dodatne informacije o izpitu.

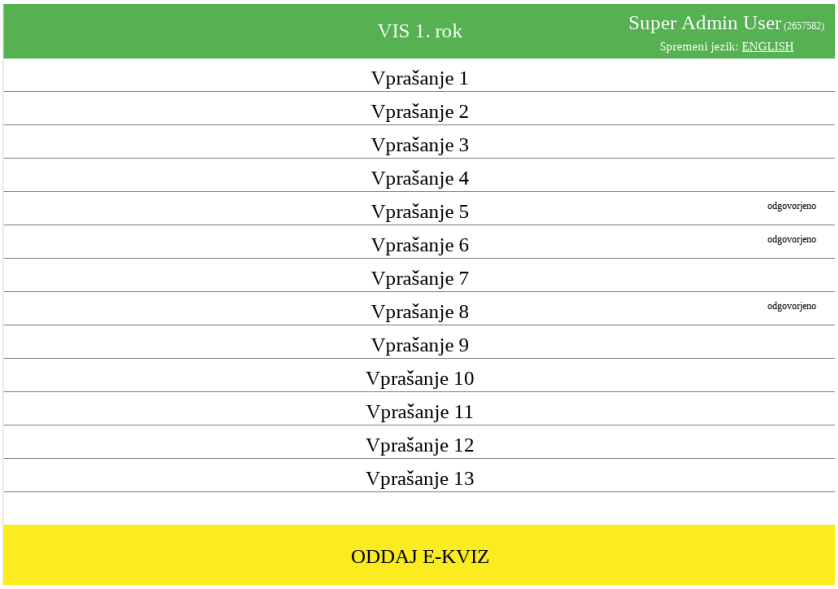
Ko se uporabnik odloči, da bo začel z reševanjem, klikne na gumb *Začni e-kviz*. Gumb je viden na sliki 3.1. Uporabnik ima na začetku izpita pregled nad vsemi vprašanji ter možnostjo oddaje oz. odjave. Izbere lahko katerokoli vprašanje, nato pa ga aplikacija preusmeri na specifično stran glede na tip izpita.

Pri nalogah imamo več vrst vprašanj.

- **essay**, uporabnik lahko vpiše poljubni tekst kot odgovor na vprašanje,
- **radio**, uporabnik se lahko odloči za natanko enega izmed podanih odgovorov,
- **checkbox**, uporabnik se lahko odloči za več različnih odgovorov, ki so na voljo,
- **numerical**, uporabnik mora kot odgovor napisati neko decimalno število zaokroženo na dve decimalni mesti.

Ko uporabnik izbere odgovor, se njegova izbira avtomatično pošlje na strežnik. Na dnu strani ima tudi pregled, koliko je še vprašanj in na katera je že odgovoril. Če se uporabnik odpravi na stran z vsemi vprašanji, lahko na desni strani vidi, katera so že bila rešena. Na ta način lahko hitro ugotovi, koliko vprašanj mu še ostaja. To se lahko vidi na sliki 3.2. Na vrhu strani ima stalno prikazane tudi svoje osebne podatke, kot so ime in priimek ter vpisna številka.

Ko se študent odloči, da je z izpitom zaključil, lahko ponovno pregleda vsa vprašanj ter nato na dnu strani klikne na gumb *Oddaj izpit*. Aplikacija ga ponovno vpraša, če se s tem strinja, in s potrditvijo se izpit zaključi. V primeru, da se strežnik med delovanjem neha odzivati, dobi uporabnik o tem obvestilo.



Slika 3.2: Na sliki so vidna vprašanja nekega izpita ter na desni strani so označena katera so bila že rešena.

MVC

Programska arhitektura, ki smo jo uporabili za izdelavo uporabniškega vmesnika, se imenuje MVC (Model-View-Controller). Ta skrbi, da so ločene funkcionalnosti kode med seboj pravilno ločene in da se na pravilen način sklicujejo ena na drugo. Tak način izdelave programov je zelo popularen pri razvoju spletnih strani, kot tudi pri izdelavi aplikacij za mobilne telefone in namizne računalnike.

Samo ime (MVC) arhitekture je razdeljeno na:

- *model*, ki skrbi za shranjevanje podatkov pridobljenih s strani krmilnika (controller) in jih nato posreduje naprej do pogleda (view),
- *view (pogled)*, vsebuje vso HTML kodo in skrbi, da se podatki, ki jih dobi od modela, pravilno prikažejo na spletni strani,
- *controller (krmilnik)*, ki skrbi za preverjanje avtentikacije uporabnika, povezovanje na bazo oz. na nek drug zunanji vir podatkov, pravilno

polnjenje modela ter izbiro pravilnega pogleda za interpretacijo podatkov.

Odzivni CSS

Za oblikovanje spletne aplikacije smo uporabili programski jezik CSS. Jezik vsebuje veliko gradnikov, s katerimi lahko oblikujemo besedila, naslove ter postavljamo elemente na želena mesta po strani. Ker imajo lahko tablice, s katerimi bomo dostopali do spletne aplikacija, različne velikosti zaslonov, je potrebno za vsako prilagoditi višino, širino ter pozicijo elementov. V ta namen imamo v CSS-ju ukaze, s katerimi definiramo, kdaj naj se nadaljnji ukazi upoštevajo. V primeru, da je zaslon večji, nekaterih elementov na strani ni potrebno zmanjšati.

Ločevanje strani z JavaScriptom

Spletna aplikacija vsebuje veliko različnih strani in vsaka izmed njih ima drugo funkcionalnost. Za ločevanje strani med seboj smo si pomagali z razredom na HTML elementu. Z JavaScript funkcijo, glej 3.3, koda razbere, za kateri tip spletne strani gre, in nato pokliče primerno funkcijo. Funkcije se med seboj razlikujejo po različnih AJAX klicih in prikazih posameznih elementov na strani.

AJAX

Tehnika programiranja, ki nam omogoča, da spletno stran posodobimo brez osveževanja celotne spletne strani, se imenuje Ajax (Asynchronous JavaScript And XML). Na kratko nam omogoča:

- posodobitev spletne strani brez osveževanja celotne strani,
- zahtevo po podatkih s strežnika - potem, ko se je spletna stran že naložila,

```
if ($("#loginWrapper").length)
  initLogin();
if ($("#selectExamWrapper").length)
  initSelectExam();
if ($("#listQuestionsWrapper").length)
  initSelectQuestion();
if ($(".multichoice").length)
  initMultiChoice();
if ($(".radio").length)
  initRadio();
```

Slika 3.3: Razlikovanje tipov spletnih strani

- prejem podatkov s strežnika - potem, ko se je spletna stran že naložila,
- pošiljanje podatkov na strežnik - v ozadju.

Medtem, ko uporabnik vpisuje odgovore na vprašanja, lahko v ozadju že pošiljamo njegove rešitev na strežnik. V primeru, da se aplikacija preneha odzivati ali pa študentu poteče čas reševanja, imamo delne podatke o izpitu že shranjene in ni strahu pred izgubo podatkov.

Registracija tablice v glavni sistem

Preden lahko začnemo z uporabo tablice, jo je potrebno aktivirati v sistem za reševanje izpitov. Ko odpremo aplikacijo na tablici, se nam pojavi okno (glej sliko 4.1) za vpis gesla. Geslo, ki ga potrebujemo za aktivacijo, se nahaja na spletnem vmesniku glavnega sistema. Ko se aplikacija odpre, se v ozadju ustvari naključen unikatni identifikator, ki se nato pošlje na strežnik. Tam nato nastavimo geslo za podan identifikator in le -tega je nato potrebno vpisati na tablico. Po končani prijavi tablice je ta na voljo za uporabo.

Tablica za vsak nadaljnji klic na strežnik zraven poda še svoj unikatni identifikator, ki je omogočen v sistemu. Če želi učitelj ali asistent tablico

oddaljeno onemogočiti, lahko enostavno v spletnem eQuiz sistemu izbriše ključ in tablica nima več dostopa do podatkov o izpitih. Po blokadi tablice jo je za ponovno uporabo potrebno prijaviti z enakim postopkom kot prvič.

Ko je aplikacija enkrat registrirana v sistem, se nam po odprtju ne prikaže stran za vpis ključa aktivacije, temveč nas takoj preusmeri na prijavno okno študenta.

Prijava študenta v sistem

Vsak študent ima možnost prijave na tablico s svojo vpisno številko ter geslom, ki ga uporablja na ostalih spletnih mestih fakultete. V primeru, da se vpisna številka in geslo študenta ujemata, ga tablica preusmeri na pregled vseh izpitov, ki jih tedaj lahko rešuje. V nasprotnem primeru se mu izpiše, da se podatki za prijavo ne ujemajo.

Študent lahko učitelja ali asistenta zaprosi za začasno geslo, ki mu bo omogočilo enkratno prijavo v sistem. Učitelj v sistem eQuiz vnese vpisno številko študenta in nato se mu pojavi začasno geslo. To geslo lahko nato študent vpiše v svoje okno za prijavo in sistem ga bo sprejel naprej. Geslo, ki ga je študent dobil, je možno uporabi samo enkrat, nato pa se iz sistema pobriše.

Komunikacija med tablico in strežnikom

Pred vsako povezavo med tablico in strežnikom se povezava dvojno preveri. Prvi del preverjanja se izvede, ko tablica za dostop do strežnika poleg zahtevka pošlje tudi svoj unikatni identifikator, ki ga je prejel pri registraciji tablice v eQuiz sistem. Strežnik na drugi strani preveri, ali je identifikator veljaven in odobren s strani profesorja ali asistenta. Po prijavi študenta v sistem se na tablico začasno shrani tudi novi naključni identifikator, ki strežniku pove, za katerega uporabnika trenutno tablica deluje. Strežnik pred odgovorom preveri tudi ta ključ in zagotovi, da ima študent res dostop do zahtevanega izpita in podatkov, ki jih želi. V primeru, da kateri izmed

```
submitToken: function (req, res) {
    var api = require('../helpers/request.js')
    var token = req.body.token;
    var code = req.body.code;

    var options = {
        path: "tablet/register",
        method: 'POST',
        data: {
            token:token,
            code:code
        }
    }
    api.getJSON(options, function (e) {
        var data = {}
        if(e.success)
        {
            res.cookie('token', e.token, { maxAge: 1 * 60 * 60 * 1000 });
            data.success = true;
            data.redirectUrl = '/login';
        }
        else{
            data.success = false;
            data.message = e.message;
        }
       return res.send(data);
    });
},
```

Slika 3.4: Na sliki lahko vidimo, da v primeru uspešne avtorizacije tablice v sistem, koda preusmeri uporabnika na prijavno stran.

ključev ni več veljaven, se študenta preusmeri na prijavno okno. Če je študent ta čas reševal izpit, se vsi njegovi delno rešeni odgovori shranijo na oddaljen strežnik in se ne izgubijo.

Na sliki 3.4 lahko vidimo del kode, ki omogoča potrjevanje tablice v sistem. Po uspešni prijavi naredimo piškot s ključem za nadaljnjo avtentikacijo tablice na strežniku. V primeru, da pride do napake pri registraciji, o tem uporabnika obvestimo.

3.3 Uporabniška izkušnja

Pri sestavljanju takšnega projekta je potrebno vnaprej dobro določiti, kje bodo katere informacije postavljene, kako se bo uporabnik pomikal po aplikaciji itd. Pri tem delu smo sodelovali s študentko NTF Evo Tjašo Jurišić, ki je sodelovalo pri projektu eQuiz. Pred začetkom dela smo se skupaj dogovorili, kako naj bi aplikacija izgledala in kakšen bo potek rabe aplikacije, od prijave pa vse do oddaje izpita.

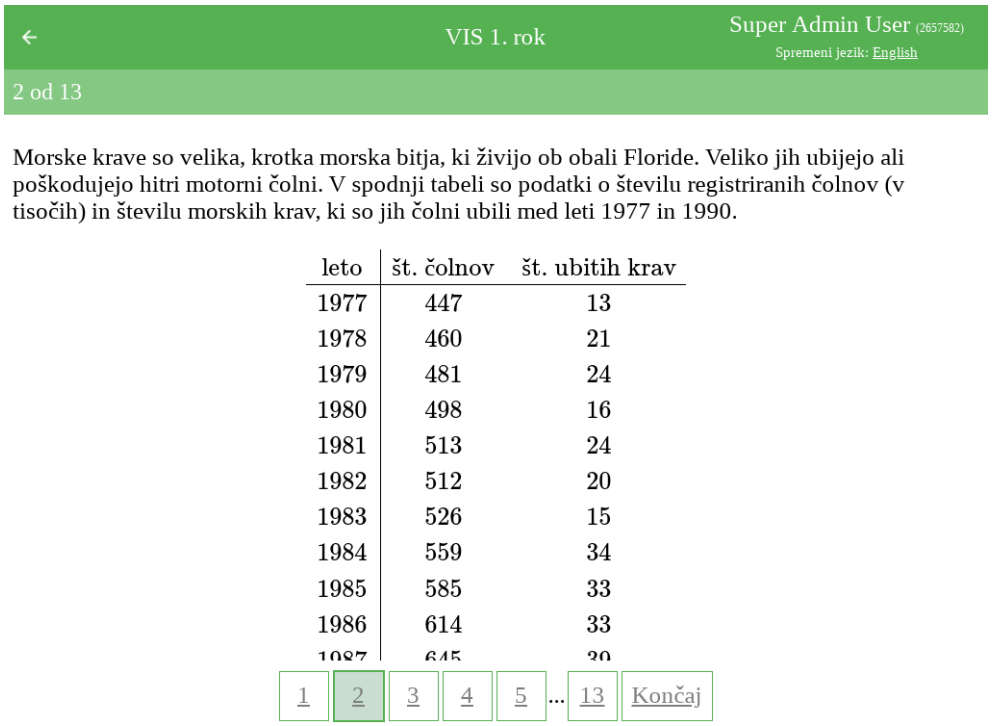
Celotna aplikacija ima 5 različnih strani:

- prijava uporabnika,
- izbira izpita za reševanje,
- pregled izpita po vseh vprašanjih,
- reševanje posameznega vprašanja na izpitu,
- oddaja izpita.

Pri vsaki strani (glej sliko 3.2) lahko študent desno zgoraj vidi tudi svoje osebne podatke, ime in priimek ter vpisno številko.

Pri prikazovanju vprašanj izpita se čez besedilo zažene tudi skripta, ki pretvori Latex kodo v grafično obliko (glej sliko 3.5). Na ta način lahko študent vidi formule zapisane v pravi in ne v navadni latex tekstovni obliki. Če je na vprašanje možnih več pravilnih odgovorov, je pred vsakim vprašanjem prikazan kvadratek, ki ponazarja, da lahko zbere več odgovorov. Ko je pravilen samo en odgovor, so prikazani krogci.

Uporabnik se lahko med vprašanji premika tudi prek paginacije na dnu strani. Če je vprašanj več kot se lahko prikaže številk na zaslonu, se prikaže samo prva ter zadnja in 3 vmesne strani. Vmesne strani se preračunajo glede na trenutno stran, na kateri se nahaja, glej sliko 3.5.



Slika 3.5: Primer, kako izgleda neko vprašanje na izpitu

Poglavje 4

Varnost

Glavni izziv pri izdelavi aplikacije, je bila varnost. Pri tem smo poskrbeli, da je njena uporaba omejena izključno na pooblaščene uporabnike. Pri tem smo dodali tudi zaščito, da končni uporabnik med reševanjem izpita ne more program zapreti in zamenjati na neko drugo aplikacijo. Poskrbeli smo tudi, da je prepisovanje rešitev oteženo, za dostop do reševanja lahko uporabi samo v naprej odobrene naprave ter onemogočili večkratnega reševanja izpitov. V nadaljevanju je podrobneje opisana vsaka funkcija, ki pripomore k zaščiti aplikacije.

4.1 Avtentikacija tablice

Vsako tablico je potrebno pred njeno prvo uporabo registrirati v sistem. Ko se program na njej prvič zažene, se ji naključno zgenerira ime. Prikaz strani lahko vidimo na sliki 4.1. Le-to administrator vidi znotraj admin vmesnika eQuiz. Vodja izpita tablico izbere in jo omogoči. V naslednjem koraku dobimo naključno generirano geslo, ki ga je potrebno še vnesti v aplikacijo. Po uspešno vnešenem in potrjenem geslu, jo lahko uporabljamo za reševanje kvizov. Priporočeno je, da se takšno operacijo izvede enkrat na leto saj s tem poostrimo pregled nad aktivnimi tablicami in pridobimo nadzor, katere tablice imajo nameščen naš sistem in katere izmed njih lahko uporabljamo za

Slika 4.1: Prikaz strani, preden je tablica avtenticirana.

izpite. Kako se odobri aplikacijo v spletnem vmesniku vodje izpita si lahko preberemo v poglavju Dodajanje dovoljenih naprav.

Tehničen del registracije v sistem je narejen na osnovi intervalno-oddajanja svojega unikatnega imena na strežnik. Ko nas aplikacija preusmeri na spletno stran za vnos ključa, se nam pred tem določi naključno ime, ki se ga posreduje na strežnik. Nato od njega dobimo odgovor, ki vsebuje parameter 'Token', ki strežniku pove, za katero napravo gre. Ko se nam spletna stran za vnos gesla naloži, se nam s pomočjo JavaScripta v ozadju pošilja zahtevek na strežnik in s tem sporočimo, da smo se aktualni za odobritev. Skripta se zažene vsakih 2000 milisekund, v zahtevku pa poleg svojega naključno generiranega imena pošlje še 'Token', ki ga je predhodno dobila od strežnika. Pri tem moramo poudariti, da v primeru, ko prenehamo z intervalnim pošiljanjem zahtevkov, se naša aplikacije skrije s prikaza vseh naprav v spletnem vmesniku.

Ko vnesemo ključ in ga potrdimo s klikom na gumb 'Registracija', se na strežnik pošlje 'Token' ter naš vnešeni ključ. Zahtevek lahko vidimo na

```
submitToken: function (req, res) {
    var api = require('../helpers/request.js')
    var token = req.body.token;
    var code = req.body.code;

    var options = {
        path: "tablet/register",
        method: 'POST',
        data: {
            token:token,
            code:code
        }
    }
    api.getJSON(options,req.customData, function (e) {
        var data = {}
        if(e.status)
        {
            //show some json:
            res.cookie('token', e.token, { maxAge: 180*24*1 * 60 * 60 * 1000 }); //180 dni
            data.success = true;
            data.redirectUrl = '/login';

        }
        else{
            data.success = false;
            data.message = e.message;
        }
       return res.send(data);
    });
}
```

Slika 4.2: V kodi lahko vidimo, kako izgleda potrditev kode, za prijavo tablice v sistem.

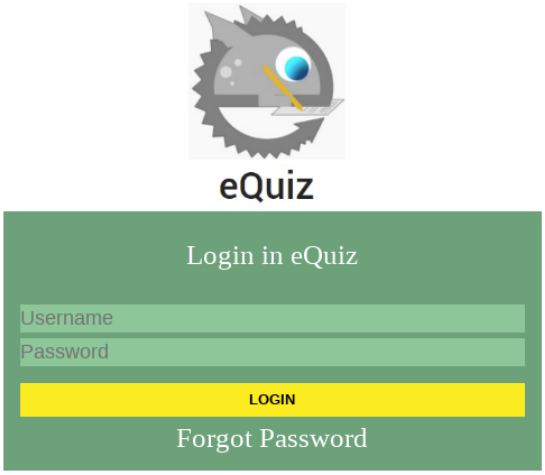
sliki 4.2. Strežnik nam odgovori s statusom, če je bila akcija izvedena uspešno ali ne. V primeru, da je prišlo do napake, nam v odgovor sporoči, kaj je bilo narobe, v nasprotnem primeru, pa nastavi Android tablici svoj unikaten 'Token', ki smo ga predhodno uporabljali v intervalnem obveščanju. Na koncu nas preusmeri na prijavno stran uporabnika in s tem je seznanjenje končano.

4.2 Prijava uporabnika

Ko se uporabnik prijavi v sistem, imamo podatke o tem, kdo je, kdaj se je prijavil in iz katere tablice se je prijavil. Na podlagi tega imamo točen nadzor nad tem, kdo je reševal katere kvize, kdaj in koliko časa mu je reševanje vzelo. S temi podatki lahko naknadno naredimo analize glede težavnosti izpitov, časa reševanja in podobno. Stran za prijavo uporabnika lahko vidimo na sliki 4.3.

Po uspešni prijavi uporabnika nam API strežnik vrne dva piškotka. Prvi je 'access_token', drugi pa 'refresh_token'. Pri vseh nadaljnjih klicih na strežnik, moramo v zahtevek dodati oba piškotka. S tem se strežnik zaveda, kdo od uporabnikov mu je zahtevo poslal. V primeru, da piškotka potečeta, nas aplikacija preusmeri na prijavno stran uporabnika. Na sliki 4.4 si lahko pogledamo, kako je narejen AJAX zahtevek na strežnik, ki kot jedro zahtevka pošlje uporabniško ime in geslo uporabnika. V primeru, da je prijava uspešna, se nastavita zgoraj omenjena piškotka, nato pa se v nadaljnji kodi to upošteva pri preusmeritvi uporabnika na izbiro izpita.

Glavna metoda pri preverjanju uporabnikovega stanja prijave je 'checkAuth' (glej sliko 4.5). Naloga metode je, da pregleda uporabnikove piškotke pri vsakem zahtevku po strani. Če uporabnik nima piškotka 'Token' to pomeni, da tablica še ni bila odobrena s strani administratorja. V tem primeru te mora preusmeriti na stran, kjer lahko skrbnik doda napravo v seznam dovoljenih. Nadaljnji pregled piškotkov pa je, da preveri, če ima uporabnik piškotka, ki jih dobi pri prijavi v sistem. Če ju ima, lahko nadaljuje na željeno stran, v



Slika 4.3: Prikaz strani za prijavo uporabnika

```
submitLogin: function (req, res) {
    var api = require('../helpers/request.js')

    var token = req.cookies["token"];
    req.body.token = token;

    var options = {
        path: "tablet/login",
        method: 'POST',
        data: req.body
    }
    api.getJSON(options, req.customData, function (e, host, refreshToken, accessToken) {
        if (e.success) {
            res.cookie('access_token', accessToken, { maxAge: 1 * 60 * 60 * 1000 });
            res.cookie('refresh_token', refreshToken, { maxAge: 1 * 60 * 60 * 1000 });
            e.success = true;
            e.redirectUrl = '/select-exam/index';
        }
        return res.send(e);
    })
},
```

Slika 4.4: Koda za prijavo uporabnika v sistem

```
checkAuth: function (req, res, next) {
    var allowedPaths = ['/Images/left.png'
                        ,'/login'
                        ,'/login/otps'
                        ,'/login/submit-login'
                        ,'/announce'
                        ,'/announce/submit-token'
                        ,'/announce/show-token']

    var authCookie = req.cookies["auth"];
    req.authenticated = authCookie !== undefined && authCookie !== ""; //add custom logic

    req.customData = {};
    req.customData.accessToken = req.cookies["access_token"];
    req.customData.refreshToken = req.cookies["refresh_token"];

    if (req.path == "/favicon.ico") {
        next();
        return;
    }
    if (!req.path.startsWith("/announce") && (req.cookies["token"] === undefined || req.cookies["token"] === "")) {
        res.redirect("/announce");
        return;
    }

    if (!req.authenticated && allowedPaths.indexOf(req.path) < 0) {
        res.redirect("/login");
        return;
    }

    next();
},
```

Slika 4.5: Izrez kode, ki skrbi za pravilno preverjanje uporabnikove prijave

nasprotnem primeru, ga preusmeri na prijavno stran.

4.2.1 Začasno geslo

Ko se uporabnik želi prijaviti v sistem, mora podati svoje uporabniško ime ter geslo, ki mu je dodeljeno. V primeru, da je uporabnik geslo pozabil in se ne more prijaviti v sistem, lahko pri nadzorni osebi izpita sproži poziv za pridobitev začasnega gesla. Ta dobi poziv v admin sekcijo eQuiza in mu lahko tam dodeli začasno geslo, ki ga nato uporabnik vnese v polje, kjer je prej vpisoval svoje geslo. Le-to je veljavno samo enkrat in se s tem ne more ponovno prijaviti. Če se želi prijaviti ponovno, mora spet prositi za začasno geslo, ki ga skrbnik znova odobri in mu ga nato ustno ali pisno sporoči.

4.3 Zapiranje aplikacije

Program, ki je zagnan na Android sistemu, deluje tako, da se v primeru, ko uporabnik aplikacijo zapre, le-ta avtomatično ponovno zažene. Na ta način je uporabniku onemogočena uporaba drugih aplikacij, medtem ko rešuje kviz. S

```
TimerTask task = new TimerTask() {
 @Override
 public void run() {

  // If you wish to stop the task/polling
  if (stopTask) {
   this.cancel();
  }

  ActivityManager activityManager = (ActivityManager) getSystemService(Context.ACTIVITY_SERVICE);
  // The first in the list of RunningTasks is always the foreground task.
  ActivityManager.RunningTaskInfo foregroundTaskInfo = activityManager.getRunningTasks(1).get(0);
  String foregroundTaskPackageName = foregroundTaskInfo.topActivity.getPackageName();

  // Check foreground app: If it is not in the foreground... bring it!
  if (!foregroundTaskPackageName.equals(YOUR_APP_PACKAGE_NAME)) {
   Intent LaunchIntent = getPackageManager().getLaunchIntentForPackage(EQUIZ_APP_PACKAGE_NAME);
   startActivity(LaunchIntent);
  }
 }
};
Timer timer = new Timer();
timer.scheduleAtFixedRate(task, 0, INTERVAL);
```

Slika 4.6: Tukaj lahko vidimo logiko, ki skrbi, da je naša aplikacija vedno odprta

tem smo preprečili, da bi lahko pridobil dostop do aplikacij, ki so namenjene za komunikacijo z drugimi uporabniki ali pa rešitve poiskal na spletu.

Na sliki 4.6 lahko vidimo kodo, ki se izvaja vsakih 300ms. Ta del kode pridobi podatke o tem, katera aplikacija se trenutno izvaja na glavnem zaslonu. V primeru, da to ni aplikacija eQuiz, program ponovno aktivira klic za zagon naše aplikacije. Zanimivo je omeniti, da se koda, izvaja samo takrat, ko je uporabnik med reševanjem izpita. V ostalih primerih, ko se nahaja v fazi prijave v sistem, ima omogočeno, da se program lahko zapre. Pri tem smo dobili možnost, da lahko uporabljamo nastavitve Android tablice za vzpostavo interneta, nadgradnjo eQuiz aplikacije in podobno. To funkcionalnost smo pridobili s pomočjo posebnega URL parametra. Ko se v URL povezavi nahaja parameter 'locktablet=true', nam pove, da uporabnik ne sme zapustiti aplikacije. Če parameter nastavimo na 'false', ali pa ga iz povezave izpustimo, uporabniku omogočimo, da aplikacijo zapre in po želji odpre katero drugo.

```
var url = wrapper.data("url");
var examId = getParameterByName("idexam");
var exerciseId = getParameterByName("idexercise");

wrapper.on('click keyup','.content input,.content textarea',$.debounce(1000,function(e){
  disableNavigation(true);
  var type = $(this).closest(".type");

  var data = {
    examId : examId,
    exerciseId : exerciseId,
    questionGUID : null,
  }

  var list = type.hasClass("radio") || type.hasClass("multichoice");
```

Slika 4.7: Primer kode za samodejno shranjevanje odgovorov

4.4 Sprotno shranjevanje odgovorov

V primeru, da tablica izgubi dostop do interneta ali pa ji zmanjka baterije, se uporabnikovi odgovori ne izgubijo, saj se po vsakem oddanem odgovoru ta avtomatično shrani na strežnik, uporabnika pa o tem tudi obvesti. Če se uporabnik v kviz prijavi na drugi tablici, lahko nadaljuje reševanje kviza od tam, kjer je prej končal. Skrbnik lahko sproti vidi koliko odgovorov so uporabniki že rešili in lahko oceni, ali je dovolj časa za zaključek kviza.

Na sliki 4.7 je primer kode, ki se izvede vsakič, ko uporabnik nekaj napiše v polje za odgovor ali pa izbere enega izmed že podanih. Na začetku funkcije, je klic globalno definirane funkcije, ki onemogoča uporabniku, da bi med shranjevanjem zapustil stran. Nato se naredi model, ki vsebuje vse podatke o izpitu, naknadno pa se mu doda še odgovore uporabnika. Ko se klic shranjevanja konča, se uporabniku ponovno omogočijo vse navigacijske enote, da lahko stran zapusti.

Poglavje 5

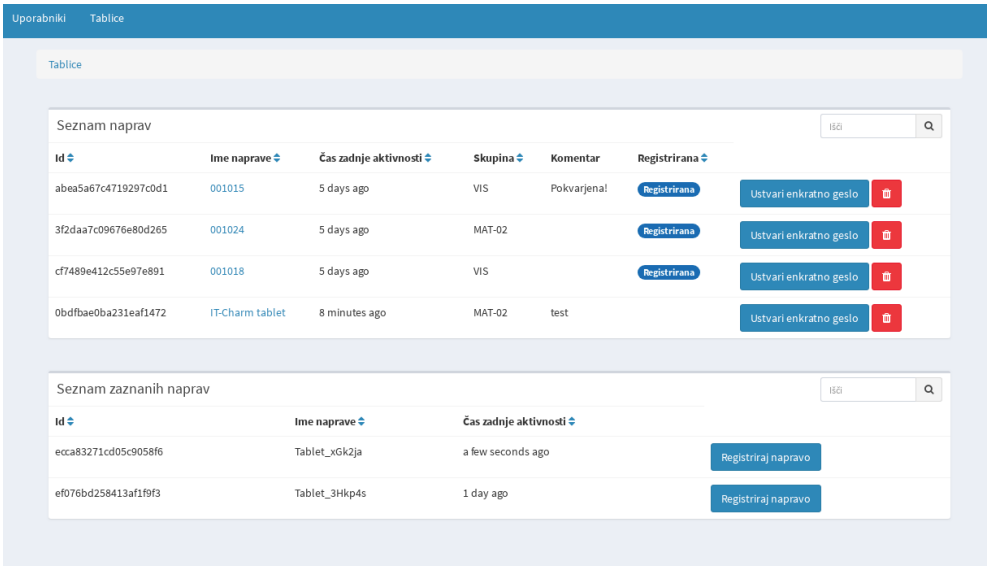
Administracija naprav in izpitov

Do sedaj smo omenili, kako izgleda aplikacija za končnega uporabnika pri reševanju izpitov in kvizov, sedaj pa naj še omenimo, kaj je počel drug del ekipe eQuiz. Njihova naloga je bila postaviti spletni vmesnik prek katerega bo lahko skrbnik pregledal aktivne Android naprave, dodajal nove v sistem ter izdeloval naloge in pregledal rezultate oddaj. Spletni vmesnik je trenutno še v izpopolnjevanju, saj bo poleg omenjenih zadev omogočal tudi reševanje izpitov, upravljanje z uporabniki in podobno.

5.1 Dodajanje dovoljenih naprav

Preden lahko uporabimo napravo za reševanje izpitov smo omenili, da moramo napravo avtenticirati v sistem. Spodaj so opisani koraki, kako to storimo.

1. Pri prvem zagonu programa eQuiz na napravi, se mu dodeli naključno generirano ime, ki ga skrbnik nato lahko vidi v eQuiz spletnem vmesniku. Po uspešni prijavi v vmesnik gremo na stran 'Tablice'. Na strani lahko vidimo dva sklopa naprav. Prvi sklop je spisek naprav, ki so že



Slika 5.1: Pregled Android naprav v sistemu eQuiz

bile odobrene s strani skrbnika, v spodnjem sklopu pa lahko vidimo na novo odkrite naprave. Na sliki 5.1 lahko ta dva sklopa tudi vidimo.

2. Če želimo omogočiti novo napravo, kliknemo na 'Registriraj napravo' v vrstici z naključnim imenom, ki ga lahko vidimo na eQuiz programu na Android tablici.

3. Po kliku nanj, se nam odpre okno (glej sliko 5.2), v katerega lahko napišemo poljuben komentar, napravo pa kategoriziramo v primerno skupino.

4. S klikom na 'Continue' se nam odpre novo okno, v katerega moramo vnesti naše geslo za prijavo v spletni vmesnik. S tem smo poostrili zaščito, da nepooblaščena oseba ne more tako enostavno dodati nove naprave.

5. Po uspešnem vnešenem geslu in klikom na gumb 'Registriraj', se nam izpiše enkratno geslo. To geslo nato vpišemo v program na tablici ter potrdimo s klikom na gumb 'Registracija'.

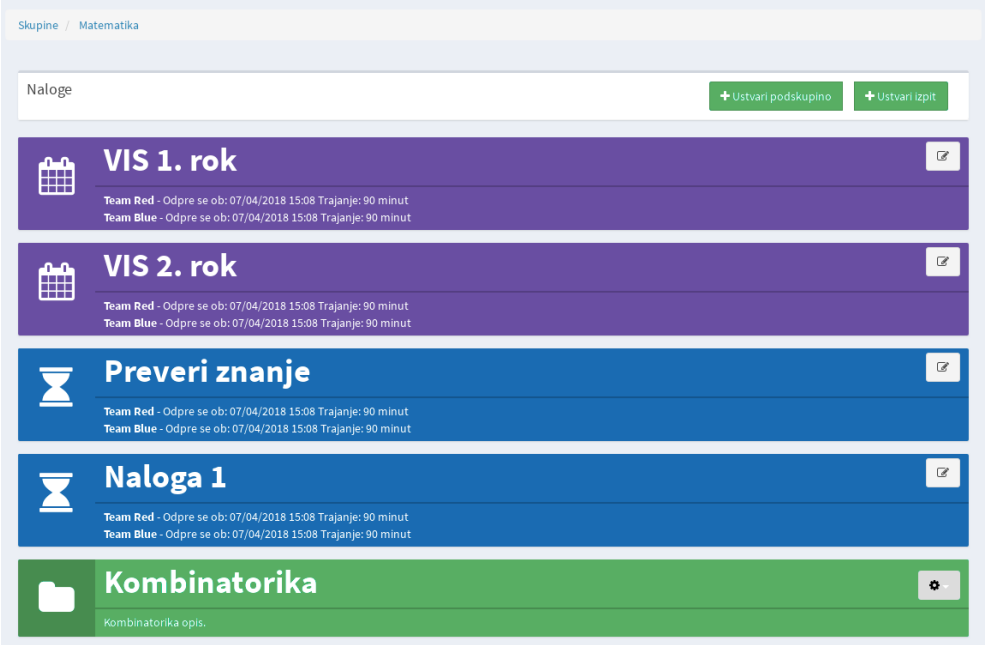
Slika 5.2: Primer okna za dodajanje nove naprave v sistem

Če je vnešeno geslo pravilno, nas preusmeri na stran, kjer se lahko sedaj študent prijavi za reševanje izpita. Na spletnem vmesniku lahko sedaj to napravo vidimo v zgornjem sklopu, poleg ostalih, že seznanjenih naprav. S klikom na ikono koša, lahko kadarkoli napravo tudi odstranimo, ta pa nemudoma izgubi dostop do API-ja.

5.2 Dodajanje izpita ali kviza

V spletnem vmesniku lahko poleg izpitov dodajamo tudi kvize. Po prijavi vanj, imamo na domači strani prikaz vseh skupin, ki jih vzdržujemo (glej sliko 5.3). S klikom na 'Ustvari skupino', lahko dodamo novo ali pa se s klikom nanjo pomaknemo nivo nižje. Ko se enkrat nahajamo v skupini, imamo možnost dodajanja novega izpita. S klikom na 'Ustvari izpit' se nam odpre pojavno okno, v katerega vnesemo novo ime izpita.

S klikom na gumb, ki se nam pojavi v pojavnem oknu za ustvarjanje



Slika 5.3: Pregled vseh izpitov in nalog, ki jih imamo v sistemu

izpita, vmesnik nastavi vse potrebne nastavitve, da lahko nato izpit še uredimo. V pregledu izpitov in kategorij, s klikom na ikono 'Urejanje' poleg izpita, lahko le-tega po želji popravljamo. Dodajamo lahko vprašanja in podvprašanja, nastavljamo možne termine reševanj ter označimo, če lahko uporabnik ta izpit rešuje večkrat.

Pri dodajanju vprašanj imamo tudi možnost dodeljevanja točk. Na ta način določimo, koliko točk uporabnik dobi za pravilen odgovor in koliko negativnih točk dobi za nepravilen odgovor. Ko zaključimo z urejanjem izpita, nam na koncu ostane samo še naloga, da nastavimo časovne termine, kdaj se izpit lahko rešuje. Enkrat, ko to nastavimo, izpita ni mogoče ponovno urejati.

Poglavje 6

Testiranje aplikacije

V svetu programiranja najrazličnejših aplikacij je ključen korak tudi testiranje. Pri programiranju aplikacij se ponavadi osredotočimo na eno napravo in jo nato, na koncu, preverimo še na ostalih. Cilj postopka testiranja pa je, da aplikacijo preverijo drugi uporabniki, na svojih napravah, saj svojih napak ne opazimo tako hitro in tudi nekatere funkcije ne delujejo na vseh operacijskih sistemih enako. Vsak tester, ki preverja funkcionalnost aplikacije, jo uporablja na malo drugačen način, kot jo je preverjal programer. Pri tem pa lahko opazimo napake, za katere sploh nismo vedeli, da lahko obstajajo. V poglavju sem opisal, kako smo ta korak izvedli pri Android aplikaciji in kakšni so bili rezultati njenega testiranja.

6.1 Načrt testiranja

Preden smo pričeli s testiranjem aplikacije, smo pripravili spisek ključnih funkcionalnosti, ki jih ta omogoča. Nato smo pripravili obrazec, na katerem so bila polja, ime in priimek, datum ter ura in uporabniško ime, s katerim bo aplikacijo preverjal. Na spodnji tabeli so napisane vse funkcije, na kar je moral biti tester pozoren.

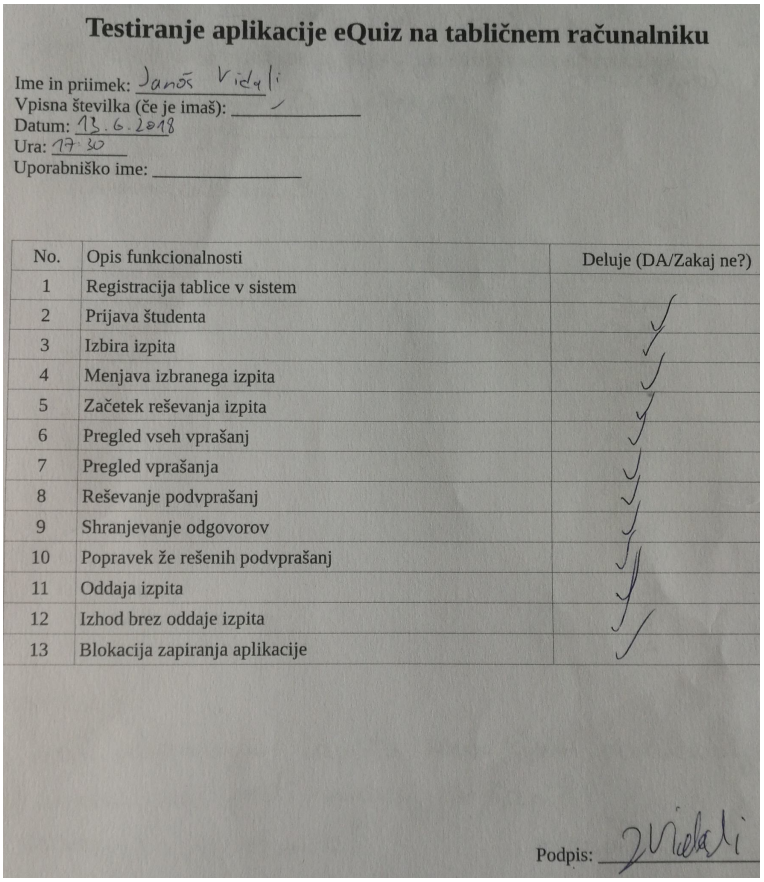
- Registracija Android tablice v sistem,
- prijava študenta,

- prijava z začasnim geslom,
- izbira izpita,
- menjava izbranega izpita,
- začetek reševanja izpita,
- pregled vseh vprašanj,
- pregled vprašanja,
- reševanje podvprašanj,
- shranjevanje odgovorov,
- popravek že rešenih podvprašanj,
- oddaja izpita,
- blokada preklopa med programi.

Cilj je bil, da bo vsak preveril vse operacije, ki jih aplikacija nudi in pri tem ne bo naletel na nobeno napako. V primeru, ko bo opazil, nekaj kar ne deluje, bo poleg funkcionalnosti napisal, kaj je narobe. Na dnu obrazca je lahko še dodal kakšno svoje mnenje o programu ali predlogo, ki si jo je spomnil med uporabo programa.

6.2 Izvedba

Testiranje aplikacije, smo izvedli v naprej določenem dnevu in uri. Skupaj smo se dobili v sejni sobi v tretjem nadstropju Fakultete za računalništvo in informatiko Ljubljana. Prišli so trije študentje, ki tudi sodelujejo na projektu eQuiz, ter en asistent. Vsak udeleženec je prejel obrazec, v katerem je sproti dopolnjeval opombe in napake aplikacije. V naprej pripravljenih dveh Android tablicah znamki Charm Intex, je bila naložena aplikacija eQuiz ter omogočen dostop do interneta.



Testiranje aplikacije eQuiz na tabličnem računalniku

Ime in priimek: Janoš Vidali
Vpisna številka (če je imaš): ✓
Datum: 13.6.2018
Ura: 17:30
Uporabniško ime: _____

No.	Opis funkcionalnosti	Deluje (DA/Zakaj ne?)
1	Registracija tablice v sistem	
2	Prijava študenta	✓
3	Izbira izpita	✓
4	Menjava izbranega izpita	✓
5	Začetek reševanja izpita	✓
6	Pregled vseh vprašanj	✓
7	Pregled vprašanja	✓
8	Reševanje podvprašanj	✓
9	Shranjevanje odgovorov	✓
10	Popravek že rešenih podvprašanj	✓
11	Oddaja izpita	✓
12	Izhod brez oddaje izpita	✓
13	Blokacija zapiranja aplikacije	✓

Podpis: J. Vidali

Slika 6.1: Izgled rešenega obrazca

Vsak se je poskusil prijaviti, izbrati izpit ter ga rešiti. Na koncu testiranja je izpit še oddal in s tem se je testiranje zanj zaključilo. Med reševanjem je poskušal spremeniti že shranjene odgovore, prehode med različnimi vprašanji in aplikacijo tudi zapreti. Ko je uporabnik program zaprl, se je le-ta nemudoma ponovno prižgal, še preden, bi lahko med tem časom prešel na internet in vzpostavil komunikacijo z ostalim, zunanjim svetom.

Na sliki 6.1 lahko vidimo, kako je izgledal rešen obrazec asistenta. S kljukicami poleg funkcionalnosti je označil, ali to deluje, tako kot mora. Če je med uporabo opazil kakšno manjšo napako, je to takoj sporočil in smo to popravili, še preden je zaključil s pregledom celotne aplikacije.

```
TypeError: /home/jure/diplomaKoda/euganke/tablet-server/views/exam/partials/radio.pug:2
    1| .type.radio
  > 2|       each answer in q.answers
    3|          .row
    4|          - result = q.submittedAnswer != null?q.submittedAnswer.answer:[]
    5|          - checked = result===answer.guid

Cannot read property 'length' of null
    at eval (eval at wrap (/home/jure/diplomaKoda/euganke/tablet-server/node_modules/pug-runtime/wrap.js:6:10), <anonymous>:347:32)
    at eval (eval at wrap (/home/jure/diplomaKoda/euganke/tablet-server/node_modules/pug-runtime/wrap.js:6:10), <anonymous>:398:4)
    at eval (eval at wrap (/home/jure/diplomaKoda/euganke/tablet-server/node_modules/pug-runtime/wrap.js:6:10), <anonymous>:667:4)
    at template (eval at wrap (/home/jure/diplomaKoda/euganke/tablet-server/node_modules/pug-runtime/wrap.js:6:10), <anonymous>:672:123)
    at Object.exports.renderFile (/home/jure/diplomaKoda/euganke/tablet-server/node_modules/pug/lib/index.js:428:38)
    at Object.exports.renderFile (/home/jure/diplomaKoda/euganke/tablet-server/node_modules/pug/lib/index.js:418:21)
    at View.exports.__express [as engine] (/home/jure/diplomaKoda/euganke/tablet-server/node_modules/pug/lib/index.js:465:11)
    at View.render (/home/jure/diplomaKoda/euganke/tablet-server/node_modules/express/lib/view.js:135:8)
    at tryRender (/home/jure/diplomaKoda/euganke/tablet-server/node_modules/express/lib/application.js:640:10)
    at Function.render (/home/jure/diplomaKoda/euganke/tablet-server/node_modules/express/lib/application.js:592:3)
```

Slika 6.2: Primer napake, pri prikazovanju vprašanja

Uporabniki, ki so testirali aplikacijo, so iskali napake, ki se pojavijo v prevodih iz angleščine v slovenščino, napake pri prikazu vprašanj, pošiljanje odgovorov na strežnik in podobno. Nekajkrat so prišli tudi do večje napake, prek katere niso mogli nadaljevati s testiranjem. Napako je bilo potrebno takoj popravili. Enega izmed takšnih problemov lahko vidimo na sliki 6.2. Vzrok za to je, da je program želel prikazati možne odgovore na vprašanje, a mu jih API ni posredoval. To smo takoj popravili tako, da smo dodali preverjanje, če možni odgovori obstajajo in uporabnik je lahko nato nadaljeval z reševanjem izpita.

6.3 Popravki

Med testiranjem smo naleteli na kar nekaj manjših napak. Pri tem govorimo o napačnih prevodih, napak pri prikazovanju in shranjevanju vprašanj, možnost večkratnega reševanja izpita in podobno. Med testiranjem je eden izmed testerjev opazil tudi napako pri gumbu, saj klik nanj ga ni preusmeril na prejšnjo stran. Napako smo takoj popravili in prikazali na produkciji.

V osnovi so se napake ločile na dva tipa. Prvi tip napak je, ko se napaka pojavi na Android tablici. V tem primeru moramo kodo popraviti, nato pa ta popravek ročno namestiti na vsako tablico posebej. Napake, kar se tiče Android tablice, se navezujejo na zapiranje aplikacije, napačno ravnanje s sejo uporabnika in podobno. Druga vrsta napak pa je, ko se pojavi na sple-

tnem strežniku. Ko to napako popravimo, kodo, ki se nahaja na strežniku, posodobimo in spremembo opazijo vsi uporabniki. Na našo srečo, se je pri nas pojavilo več takšnih napak in smo to lahko takoj popravili.

Četudi smo imeli na začetku občutek, da je celotna aplikacija majhna ter brez napak, smo jih po celovitem skupnem testiranju opazili kar nekaj. S tem, ko smo to popravili, lahko potrdimo, da je program spraven in se te napake v prihodnje ne morejo več pojaviti. Od sedaj naprej, lahko pri dodajanju novih funkcionalnostih preverimo samo novi del, saj za ostali, stari del, že vemo, da deluje brezhibno.

Poglavje 7

Izvajanje izpita na tablicah

Po nekaj tednih testiranja in pregledovanja, če aplikacija deluje brezhibno, smo se le na koncu odločili, da 3. izpitni rok izvedemo na tablicah. Preden smo obvestili študente o tem, kako bodo lahko reševali ta izpit, smo morali temeljito pregledati podatke, ki jih imamo v bazi. Pregledali smo uporabnike, izpite ter na koncu poskusili nekaj testnih izpitov rešiti tudi sami. V nadaljnjih poglavjih, smo opisali, kako nam je uspelo speljati izvedbo izpita, brez večjih težav in zapletov.

7.1 Priprave na izpit

Vedno pravijo, da priprava na neko nalogo vzame več časa, kot pa nato sama izvedba. Tako je bilo tudi pri nas. Preden smo bili pripravljeni za izvedbo izpita, smo morali postaviti ločen strežnik, ki bo skrbel za komunikacijo s tablico, kot tudi za prikaz administratorskega vmesnika za dodajanje ter urejanje izpitov.

Po namestitvi strežnika, smo prekopirali uporabnike iz starega eQuiz sistema v novega. S tem smo zagotovili, da so uporabniki na voljo v novem sistemu brez predhodne registracije. Za prijavo so potrebovali svoje staro uporabniško ime in geslo. Po uspešnem uvozu uporabnikov, je bil čas, da se posvetimo izpitu. Profesor je preko administratorskega vmesnika naložil .tex

datoteko, ki je vsebovala vse podatke o izpitu. Pri uvozu datoteke, smo imeli nekaj težav saj struktura, ki je bila zapisana v datoteki, ni bila prepoznana s strani strežnika. Po nekaj dodanih popravkih, nam je na koncu le uspelo.

Sedaj smo imeli pripravljene uporabnike in izpit. Potrebno je bilo samo še urediti vse tablice. Pred izpitom, smo od mentorja prejeli 40 tablic znamke Charm Intex (omenjene v poglavju 3.1), na katere smo naložili program za zagon eQuiz sistema. Vsako tablico smo povezali na internet, namestili program, nato pa je sledila še avtentikacija v sistem. V prejšnjem poglavju (glej 5.1) smo opisali, da je potrebno vsako tablico pred prvo uporabo v administratorskem sistemu odobriti, kjer urednik tablic dobi enkratno geslo, ki ga more nato vpisati na tablico. Ker smo si želeli postopek malo poenostaviti in pohitriti, smo začasno nastavili eno naključno geslo, ki smo ga lahko uporabili pri vseh tablicih.

Vsaka tablica, ki se poveže v sistem, si pred tem določi naključno ime. Na računalniku smo si naredili tabelo in vanjo vpisovali vsa naključno generirana imena. V naslednji stolpec smo poleg imena zapisali še inventarno številko tablice, ki se nahaja na zadnji strani. S tem smo želeli imeti preverjanje, katera tablica je odobrena in morebitne napake na njej dopisati v naslednji, tretji, stolpec. Pri nameščanju programa smo opazili, da ima ena naprava nameščen prestar operacijski sistem, ki ni kompatibilen z eQuiz sistemom. Ena izmed naprav pa je imela manjše napake na zaslonu, a ne tako motoče, da ne bi bila primerna za izvajanja izpita. Po nekem času smo naleteli še na eno napako na napravi, pri kateri je bila uničena baterija. Tej tablici je baterija zdržala samo nekaj minut uporabe, nato pa se je ugasnila. To smo zapisali v našo tabelo, da taka naprava ni primerna za uporabo. Ko smo vsem namestili sistem ter jih ročno preverili, smo ugotovili, da imamo 38 primernih tablic za izvajanje izpita.

7.2 Izvajanje izpita

Na koncu je le prišel dan, ko smo se skupaj s študenti zbrali v veliki predavalnici in bili pripravljeni za izvedbo izpita. Pripravili smo si seznam vseh študentov, ki so se prijavili na izpit, nato pa so se po prejemu tablice nanj podpisali in zraven podpisa še prepisali inventarno številko tablice.

Na izpitu je bilo 26 študentov. Od tega sta dva študenta pozabila svoje uporabniško ime ter geslo in sta morala izpit reševati na papirju. Vsak študent, ko je na tablici začel z reševanjem, je imel 90 minut časa, da izpit odda. Če ga v tem času ni oddal, ga je sistem avtomatsko zaključil in uporabnika odjavil. Pri tem je zanimivo vedeti, da je vsak študent imel svoj rok odaje, saj je za vsakega uporabnika tekla svoja ura 90 minut.

Med izvajanjem izpita so imeli nekateri študentje pripombo, da na tablici ni označeno, koliko časa jim še ostaja za oddajo. Ker so vsi začeli z reševanjem približno ob istem času, smo jim lahko okvirno povedali, do kdaj imajo čas. Pripombe, ki so jih imeli smo si zapisali in jih nato, po izpitu, pregledali in morebitne ideje tudi dodali v sistem.

Ko je uporabniku čas potekel, ali pa je sam, predhodno, oddal izpit, je tablico odnesel do asistenta in tako so vsi študentje uspešno zaključili z izvajanjem izpita na tablicah.

7.3 Ugotovitve in analiza

Izvajanje takšnega izpita je lahko kar strestno. Vedno lahko pride do nepričakovanih napak, ki jih ne moreš takoj urediti. Dobra priprava na izpit, je obrodila sadove. Med izvajanjem ni prišlo do problemov. Prilagodili smo še sistem, da je omogočal izvoz rezultatov v csv (Comma Separated Value) obliko, da je lahko profesor ocenil odgovore študentov. Med izvajanjem izpita smo opazili, da bi lahko dodali tudi podatek o nivoju baterije. To smo naknadno tudi uredili, da imajo na levi strani zgoraj, prikazan procent baterije, ki jo tablica ima še na voljo. V primeru, ko nivo baterije pade pod 15%, se mu le-ta obarva rdeče.

Ko so nam študentje, po izpitu, vrnili tablice, smo preverili, koliko baterije jim je še ostalo. Po ocenitvi smo ugotovili, da se je na nekaterih tablicah baterija izprazina na 80%, v skrajnih primerih, je nekaterim tablicam ostalo le okoli 10% baterije. Lahko rečemo, da smo imeli pri tem kar nekaj sreče, da noben izmed študentov, ni izpraznil baterije do konca.

Na koncu lahko rečemo, da smo s tem enim izpitom privarčevali 24 listov, rezultati teh odgovorov, pa so bili takoj popravljeni in pri tem nismo porabili časa profesorjev in asistentov. V nadeljevanju pričakujemo, da bomo na tak način reševali, ne samo izpite, temveč tudi kvize med predavanji ali med-letna študentska preverjanja.

Poglavje 8

Sklepne ugotovitve

Diplomska naloga je obsegala programiranje v različnih programskih jezikih. Poleg Android aplikacije je bilo potrebno sestaviti tudi spletno stran in vzpostaviti komunikacijo z oddaljenim strežnikom. Pri izdelavi naše aplikacije smo sodelovali z ostalimi člani, ki so sodelovali pri projektu eQuiz [7]. A. Dobnikar [1] je postavil glavni strežnik in omogočil API dostop do podatkov, ki sem ga potreboval za izdelavo Android programa.

Izdelava diplomske naloga je bila zelo zanimiva, ne samo zaradi uporabe veliko različnih programskih jezikov, ampak tudi zaradi dela na Linux in Windows operacijskih sistemih. Razvoj Android aplikacije je potekal na emulatorjih Android telefonov v operacijskem sistemu Windows. Vseeno je na tem mestu priporočena previdnost, saj emulatorji ne prikažejo vedno programov na enak način, kot so prikazani na pravih napravah. Ko smo aplikacijo zaključili, smo morali zato še na pravih tabličnih napravah preveriti, ali je skladna s tem, kar smo videli na računalniku.

Pri razvoju sem sodeloval v skupini in pri tem sem spoznal, kako pomembno je sodelovanje z ostalimi razvijalci v projektu. Skupaj smo prilagodili API dostop, da je vračal tiste zadeve, ki sem jih potreboval za delovanje programa.

V primeru, ko vodja izpita opazi napako v nalogi, lahko s pomočjo eQuiz aplikacije enostavno in hitro popravi napake, študentje pa te popravke takoj

vidijo na svoji Android tablici. Takšni sistemi, kot je eQuiz, lahko poleg onemogočanja prepisovanj nudijo tudi nastavljanja časovnih omejitev izpitov, reševanja različnih izpitov hkrati in beleženja časovne statistike, koliko časa porabijo študentje na posameznih vprašanjih in kakšno je razmerje uspešno rešenih vprašanj.

Parekh, Dusane, Thakur in Sonawane [2] so opisali, kako so na Floridi prešli iz običajnih papirnih oblik izpita na tablično reševanje. Omenili so, da je bil njihov glavni razlog za prehod na tablice, z vidika okoljevarstva. V članku so napisali, da je njihova fakulteta na letnik potrebovala okoli 30000-40000 papirjev, za izpite, naloge ter kvize. S prehodom na tablice, bi poleg varčevanja omogočili tudi boljšo zaščito pred prepisovanjem med izpiti. Potrebno je še poudariti, da so naloge, ki iih imajo v sistemu, kriptirane. S tem so onemogočili, da bi študentje imeli naloge že pred izpitom. V članku na spletni strani [21] si lahko še preberemo, kako so prehod na tablici že storili v Indiji.

Literatura

[1] A. Dobnikar, Zaledni del eQuiz aplikacije, Diplomska naloga, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, v izdelavi (2018).

[2] N. Parekh, K. Dusane, S. Thakur, A. Sonawane, Virtual Subjective Examination on Tablets, *International Journal of Computer Science and Information Technologies* **6**/5 (2015), 4689–4691

[3] Ajax. Dosegljivo: https://en.wikipedia.org/wiki/Ajax_(programming). [Dostopano 5.1.2018]

[4] Android. Dosegljivo: https://en.wikipedia.org/wiki/Android_(operating_system). [Dostopano 22. 11. 2017].

[5] C Sharp. Dosegljivo: https://en.wikipedia.org/wiki/C_Sharp_(programming_language). [Dostopano 22. 11. 2017].

[6] Cascading Style Sheets. Dosegljivo: https://en.wikipedia.org/wiki/Cascading_Style_Sheets. [Dostopano 15. 11. 2017].

[7] e-Uganke. Dosegljivo: http://lkrv.fri.uni-lj.si/equiz/index.html [Dostopano 01. 06. 2018].

[8] Git. Dosegljivo:
https://en.wikipedia.org/wiki/Git. [Dostopano 15. 11. 2017].

[9] HTML1. Dosegljivo:
https://en.wikipedia.org/wiki/HTML1. [Dostopano 15. 11. 2017].

[10] Java. Dosegljivo:
https://en.wikipedia.org/wiki/Java_(programming_language). [Dostopano 21. 11. 2017].

[11] JavaScript. Dosegljivo:
https://en.wikipedia.org/wiki/JavaScript. [Dostopano 18. 11. 2017].

[12] JQuery. Dosegljivo:
https://en.wikipedia.org/wiki/JQuery. [Dostopano 18. 11. 2017].

[13] Koala. Dosegljivo:
http://koala-app.com/. [Dostopano 21. 12. 2017].

[14] Microsoft Visual Studio. Dosegljivo:
https://en.wikipedia.org/wiki/Microsoft_Visual_Studio. [Dostopano 25. 11. 2017].

[15] MVC. Dosegljivo:
https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller. [Dostopano 20. 12. 2017].

[16] Ram management on Android. Dosegljivo:
https://www.androidpit.com/ram-management-on-android [Dostopano 01. 06. 2018].

[17] Tablet. Dosegljivo:
http://www.imei.info/phonedatabase/16573-intex-charm-7/. [Dostopano 20. 12. 2017].

[18] tEXAM. Dosegljivo: https://www.ucan-assess.org/cms/tools/tablet-based-written-exams/. [Dostopano 01. 04. 2018].

[19] Tržni delež IIS. Dosegljivo: https://news.netcraft.com/archives/2017/02/27/february-2017-web-server-survey.html [Dostopano 01. 06. 2018].

[20] UNC. Dosegljivo: https://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/c65e97d7-a085-4d8f-bdd7-1e6a9f223708.mspx. [Dostopano 21.1.2018]

[21] With colleges testing e-tablets, exams likely to go paperless . Dosegljivo: https://yourstory.com/2017/10/examinations-paperless-e-tablets/ [Dostopano 01. 06. 2018].